Turing Machines continued

CSE 105 Week 7 Discussion

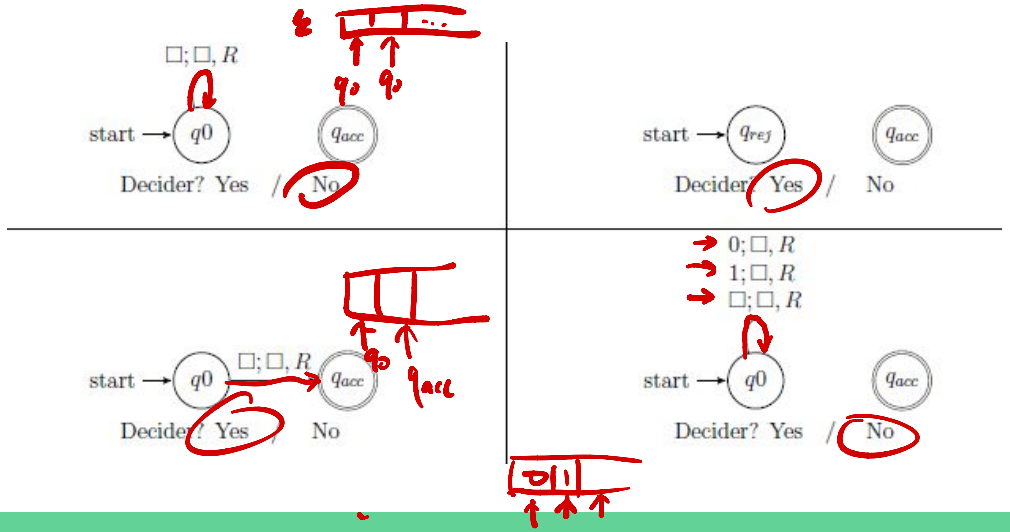
Deadlines and Logistics

- Review Test 1 score, schedule attempt 2
- Do review quizzes on PrairieLearn
- HW 5 due 2/27 (Thursday) at 5pm

Turing-recognizable and Turing-decidable

- Deciders are Turing machines that halt on all inputs; they never loop; they always make a decision to accept or reject
- Call a language Turing-recognizable if some Turing machine recognizes it
- Call a language Turing-decidable if some decider decides it

Toy examples for recap:



Multiple descriptions

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

Multiple descriptions

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

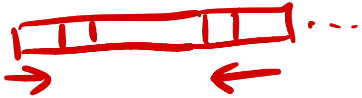
- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.
- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

L={w | w is a palindrome over {0,1} and contains an equal number of 0s and 1s}.

- Implementation Level:
- High Level:

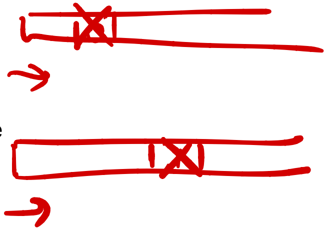
L={w | w is a palindrome over {0,1} and contains an equal number of 0s and 1s}.

- Implementation Level:



“On input w:

1. **Check palindrome:**
   a. Move to the first non-marked symbol from the left and mark it (e.g., 0 → X or 1 → Y).
   b. Move right to the end of the input (empty space), and move left to find the corresponding symbol.
      - If it matches, mark it.
      - Otherwise, *reject*.
   c. Return to the leftmost non-marked symbol and repeat until all symbols are marked or matched.
2. **Check symbol balance:**
   a. Scan the tape and mark the first 0 that has not been marked. If no unmarked 0 is found, go to stage d. Otherwise, move the head back to the front of the tape.
   b. Scan the tape and mark the first 1 that has not been marked. If no unmarked 1 is found, *reject*.
   c. Move the head back to the front of the tape and go to stage a.
   d. Move the head back to the front of the tape. Scan the tape to see if any unmarked 1s remain.
      - If none are found, *accept*.
      - Otherwise, *reject*.”



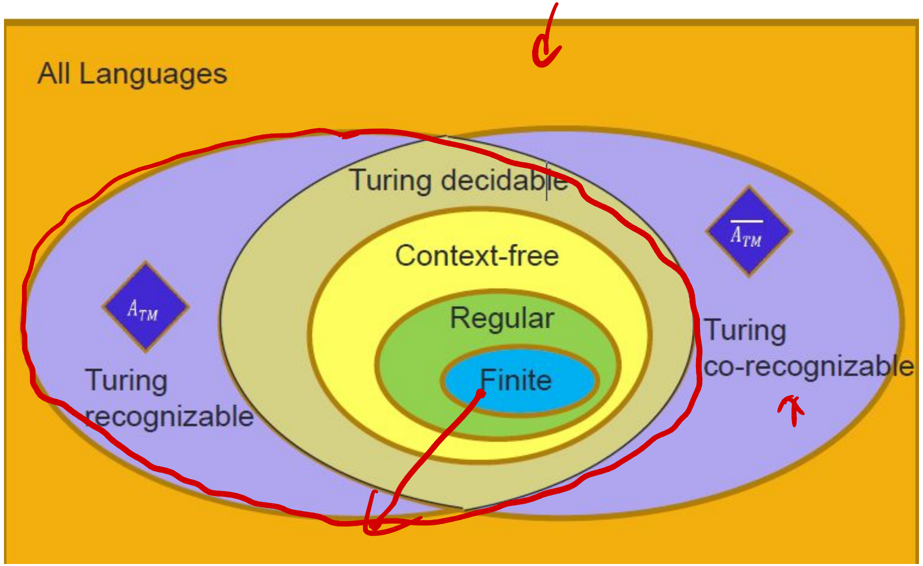
- High Level:

“On input w:

1. **Check palindrome:** Verify if w reads the same forward and backward. If not, *reject*.
2. **Check symbol balance:** Count the number of 0s and 1s in w.
   - If they are equal, accept.
   - Otherwise, reject.”

Properties of languages

1. Regular
    a. Recognized by a DFA/NFA
    b. Described by a regex
2. Context free
    a. Recognized by a PDA
    b. Generated by a CFG
3. (Turing) Decidable
    a. Can be decided by a Tm
4. (Turing) Recognizable
    a. Can be recognized by a Tm



Algorithm computation

Church-Turing Thesis

Anything that is **computable** is computable with a **Turing machine** because any method of computation using finite time and finite resources will be **equally expressive** to that of a Turing machine.

Representations of algorithms

To decide these problems, we need to represent the objects of interest as **strings**

For inputs that aren't strings, we have to **encode the object** (represent it as a string) first

To define TM M:

"On input w …

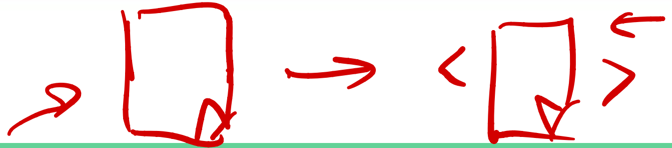
1. ..
2. ..
3. …

Notation:

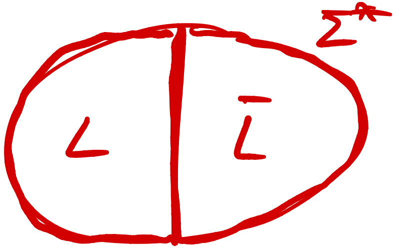
<O> is the **string** that represents (encodes) the object O

<O_1, …, O_n> is the single string that represents the list of objects O_1, …, O_n

M <M>



Turing Decidable Languages - Recap



1. A language is decidable if and only if it is co-recognizable and recognizable.
2. If two languages over a fixed alphabet are turing-decidable, then their union is decidable as well
3. If two languages over a fixed alphabet are turing-recognizable, then their union is recognizable as well

Closure Properties from Textbook

3.15 Show that the collection of decidable languages is closed under the operation of

[A]**a.** union.

b. concatenation.

c. star.

d. complementation.

e. intersection.

3.16 Show that the collection of Turing-recognizable languages is closed under the operation of

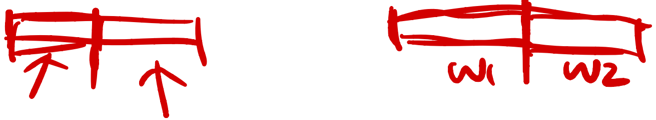
[A]**a.** union.

b. concatenation.

c. star.

d. intersection.

~~**e.** homomorphism.~~



Prove: Decidable languages are closed under concatenation

$\forall$ decidable L_1, L_2, $L' = L_1 \cdot L_2$ is also decidable.

Suppose M_1, M_2.

M': "on input w:

1. for each split $w_1 w_2$ of w:
   - a. Run M_1 on w_1. If reject, continue.
   - b. Run M_2 on w_2. If accept, accept.
2. If all options don't accept, reject.

w_1	w_2
ε	aba
a	ba
ab	a
aba	ε

aba →

$n \to n+1$

$L(M') = L'$

M' is a decider.

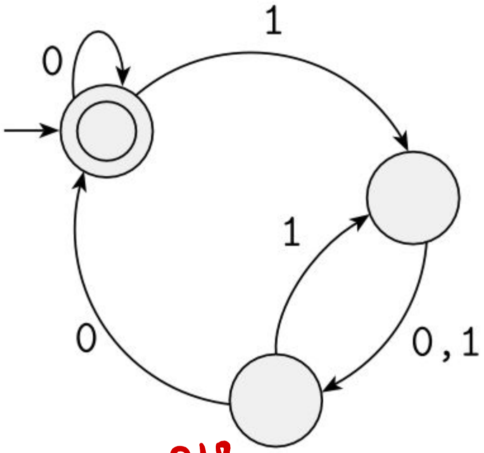
Wednesday's "lecture"...

Computational problems:

Acceptance problem		
...for DFA	A_{DFA}	$\{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}$
...for NFA	A_{NFA}	$\{\langle B, w\rangle \mid B \text{ is a NFA that accepts input string } w\}$
...for regular expressions	A_{REX}	$\{\langle R, w\rangle \mid R \text{ is a regular expression that generates input string } w\}$
...for CFG	A_{CFG}	$\{\langle G, w\rangle \mid G \text{ is a context-free grammar that generates input string } w\}$
...for PDA	A_{PDA}	$\{\langle B, w\rangle \mid B \text{ is a PDA that accepts input string } w\}$
Language emptiness testing		
...for DFA	E_{DFA}	$\{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$
...for NFA	E_{NFA}	$\{\langle A\rangle \mid A \text{ is a NFA and } L(A) = \emptyset\}$
...for regular expressions	E_{REX}	$\{\langle R\rangle \mid R \text{ is a regular expression and } L(R) = \emptyset\}$
...for CFG	E_{CFG}	$\{\langle G\rangle \mid G \text{ is a context-free grammar and } L(G) = \emptyset\}$
...for PDA	E_{PDA}	$\{\langle A\rangle \mid A \text{ is a PDA and } L(A) = \emptyset\}$
Language equality testing		
...for DFA	EQ_{DFA}	$\{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$
...for NFA	EQ_{NFA}	$\{\langle A, B\rangle \mid A \text{ and } B \text{ are NFAs and } L(A) = L(B)\}$
...for regular expressions	EQ_{REX}	$\{\langle R, R'\rangle \mid R \text{ and } R' \text{ are regular expressions and } L(R) = L(R')\}$
...for CFG	EQ_{CFG}	$\{\langle G, G'\rangle \mid G \text{ and } G' \text{ are CFGs and } L(G) = L(G')\}$
...for PDA	EQ_{PDA}	$\{\langle A, B\rangle \mid A \text{ and } B \text{ are PDAs and } L(A) = L(B)\}$

Exercise

Answer all parts for the following DFA M and give reasons for your answers.



a. Is $\langle M, 0100 \rangle \in A_{\text{DFA}}$?

b. Is $\langle M, 011 \rangle \in A_{\text{DFA}}$?

c. Is $\langle M \rangle \in A_{\text{DFA}}$?

d. Is $\langle M, 0100 \rangle \in A_{\text{REX}}$?

e. Is $\langle M \rangle \in E_{\text{DFA}}$?

f. Is $\langle M, M \rangle \in EQ_{\text{DFA}}$?

Review: A_{DFA} is a decidable language

Sipser Theorem 4.1

Review: A_{DFA} is a decidable language

1. What is A_{DFA}? Example strings?

Review: A_{DFA} is a decidable language

2. How to prove decidability?

   a. Construct a Turing Machine M
   b. Prove M is a decider
   c. Prove L(M) = A_{DFA}

Review: A_{DFA} is a decidable language

Prove: A_{NFA} is a decidable language